United States Patent Application

of

Prithwish Basu

and

Jason Keith Redi

For

SYSTEMS AND METHODS FOR AUTOMATICALLY PLACING NODES IN AN AD HOC NETWORK

SYSTEMS AND METHODS FOR AUTOMATICALLY PLACING NODES IN AN AD HOC NETWORK

GOVERNMENT CONTRACT

[0001] The U.S. Government may have a paid-up license in this invention and the right in limited circumstances to require the patent owner to license others on reasonable terms as provided for by the terms of Contract No. DASG60-02-C-0060 awarded by the Defense Advanced Research Projects Agency (DARPA).

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] The present invention relates generally to wireless networks and, more particularly, to systems and methods for automatically placing or moving nodes in a wireless network to provide biconnectivity.

Description of Related Art

[0003] Advances in electronics and mechanics have provided the basic technologies required for sophisticated robots. It is well recognized that robots have significant operational advantages over humans because they can perform tasks without requirements for rest, food, shelter, or task heterogeneity. This makes them potentially useful in future military exercises on the battlefields (so much so that they may end up undertaking all of the missions their human counterparts perform today), in several disaster relief situations (search and rescue), in cleaning cavities and surfaces that are otherwise cumbersome to clean, in collection of soil and samples on the surface of Mars (distributed sensing), in undertaking routine tasks in flexible manufacturing

environments and supermarkets, and in many other scenarios. Most of the aforementioned tasks need collaboration among different robot units for their timely and efficient completion.

[0004] Robotics researchers have proposed the use of centralized robotic networks, where all members of a team of robots communicate with a central controller (e.g., base station) over a wireless medium. In most application scenarios, such as the ones described in the previous paragraph, it is difficult to guarantee the presence of a wireless base station that can coordinate the flow of information between any two robot units. Moreover, the movement of robots can be severely restricted in order to keep in communication range of the base station. This can hamper the task that the robot team plans to execute.

[0005] There has arisen a need for self-forming, self-healing, and self-organizing multihop communications networks capable of use with autonomous and semi-autonomous robotic systems. Although numerous ad hoc network protocols, such as packet radio, mobile ad hoc networks (MANETs), or self-organizing networks, have been proposed and implemented, all of them were designed to be completely transparent to applications. One of the main reasons for adopting this approach is that the protocols are intended to be used with a wide variety of platforms and applications. The resulting extended applicability, however, comes at the cost of severe restrictions in the exchange of information between the application and the network, making it virtually impossible for them to anticipate each other's behavior and, thus, cooperate.

[0006] In robotic systems, cooperation among the robotic applications is highly desirable because robotic applications generally entail movement, which directly affects the communication network. Conversely, the propagation of radio transmissions used for

communication may be able to provide an additional way of sensing the environment. Such interaction is a feasible proposition because robots are unique in their integrated design in that the mission control, motion control, and networking protocols are typically all implemented within the same architecture.

[0007] Ad hoc networks that include robotic nodes have a salient difference from standard MANETs. In the former networks, for example, the position and motion of nodes is controllable from other nodes in the network. In the latter networks, motion is determined by the owner of the node and is not usually controllable.

[0008] As a result, there continues to be a need for self-forming, self-healing, and self-organizing multihop communications networks capable of use with autonomous and semi-autonomous robotic systems.

SUMMARY OF THE INVENTION

[0009] Systems and methods consistent with the present invention place or alter the position of robotic nodes in order to achieve biconnectivity within an ad hoc network. A subset of the nodes may be moved from their initial locations to a new set of locations such that the newly formed network is more tolerant of node failures.

[0010] In one aspect consistent with the principles of the invention, a method for achieving biconnectivity in a non-biconnected network is provided. The non-biconnected network includes multiple nodes. The method may include identifying one or more of the nodes to move and determining direction and distance to move the one or more nodes. The method may also include

moving the one or more nodes in the determined direction and distance to transform the non-biconnected network to a biconnected network.

[0011] According to another aspect, a method for achieving biconnectivity in a network that includes multiple nodes is provided. The method may include forming blocks from groups of one or more of the nodes in the network, selecting one of the blocks as a root block, and identifying other ones of the blocks as leaf blocks. The method may also include moving one or more of the leaf blocks to make the network biconnected.

[0012] According to yet another aspect, a system for achieving biconnectivity in a network that includes multiple nodes is provided. The system may include means for grouping subsets of the nodes into blocks and means for identifying cutvertices in the network. The system may also include means for iteratively moving one or more of the blocks to remove the cutvertices from the network.

[0013] According to a further aspect, at least one node in a network that includes multiple nodes is provided. The node(s) may include a network device that is capable of moving within the network and a movement controller that is configured to generate a current view of the network. The movement controller is further configured to form blocks from groups of one or more of the nodes in the network based on the current view of the network and identify one or more of the blocks to move to make the network biconnected.

[0014] According to another aspect, a method for achieving biconnectivity in a network that includes multiple nodes is provided. The method may include generating a graph of the network and identifying cutvertices in the network. The method may further include moving one or more

of the nodes in the network to systematically remove the cutvertices from the network and form a biconnected network.

[0015] According to yet another aspect, a method for achieving biconnectivity in a non-biconnected network that includes multiple nodes is provided. The method may include determining a geographic center of the non-biconnected network and moving each of one or more of the nodes a weighted distance towards the geographic center to transform the non-biconnected network to a biconnected network.

[0016] According to a further aspect, a system for achieving biconnectivity in a non-biconnected network that includes multiple nodes is provided. The system may include means for identifying a geographic center of the non-biconnected network based on current locations of the nodes, and means for causing each of one or more of the nodes to move towards the geographic center to transform the non-biconnected network to a biconnected network.

[0017] According to another aspect, a method for achieving biconnectivity in a one-dimensional non-biconnected network that includes multiple nodes is provided. The method may include determining initial positions of the nodes in the one-dimensional non-biconnected network and determining a movement schedule for the nodes using one or more linear programming techniques. The method may also include causing one or more of the nodes to move based on the determined movement schedule to form a biconnected network from the one-dimensional non-biconnected network.

[0018] According to yet another aspect, a system for achieving biconnectivity in a one-dimensional non-biconnected network that includes multiple nodes is provided. The system may

include means for determining initial positions of the nodes in the one-dimensional non-biconnected network and means for determining a movement schedule optimally in polynomial time (a function of a mathematical polynomial or series expansion) based at least in part on the initial positions of the nodes and the number of nodes in the one-dimensional non-biconnected network. The system may also include means for causing one or more of the nodes to move based on the determined movement schedule to achieve biconnectivity in the one-dimensional non-biconnected network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate the invention and, together with the description, explain the invention. In the drawings,

[0020] Fig. 1 is a block diagram of an exemplary network in which systems and methods consistent with the principles of the invention may be implemented;

[0021] Fig. 2 is an exemplary block diagram of one of the nodes in the network of Fig. 1 according to an implementation consistent with the principles of the invention;

[0022] Figs. 3A and 3B illustrate a simple example of achieving a biconnected network according to implementations consistent with the principles of the invention;

[0023] Fig. 4 is a flowchart of exemplary processing for achieving biconnectivity in a one-dimensional network according to an implementation consistent with the principles of the invention;

[0024] Figs. 5A and 5B are exemplary diagrams of a one-dimensional network according to an implementation consistent with the principles of the invention;

[0025] Fig. 6 is a flowchart of exemplary processing for achieving biconnectivity in a two-dimensional network using a contraction technique, according to an implementation consistent with the principles of the invention;

[0026] Figs. 7A-7C are exemplary diagrams of a two-dimensional network according to an implementation consistent with the principles of the invention;

[0027] Fig. 8 is a flowchart of exemplary processing for achieving biconnectivity in a two-dimensional network using a block movement technique, according to an implementation consistent with the principles of the invention;

[0028] Figs. 9A and 9B are exemplary diagrams of a two-dimensional network according to an implementation consistent with the principles of the invention;

[0029] Fig. 10 illustrates an exemplary situation where a block's movement causes an oscillation between two points;

[0030] Figs. 11A-11D illustrate an exemplary execution of a MakeBiconnected process on a randomly selected initial network topology;

[0031] Figs. 12A and 12B are exemplary graphs that illustrate performance of the one-dimensional technique;

[0032] Figs. 13A and 13B are exemplary graphs that compare the performance of the block movement technique against the contraction technique in a simulated network;

[0033] Fig. 14 is an exemplary graph of an average number of biconnected components versus the number of nodes in a simulated network;

[0034] Fig. 15 is an exemplary graph of an average number of iterations needed to achieve biconnectivity in a simulated network; and

[0035] Fig. 16 is an exemplary graph of an average diameter of a simulated network while varying the number of nodes in the simulated network.

DETAILED DESCRIPTION

[0036] The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings may identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims and equivalents.

[0037] Autonomous and semi-autonomous mobile multi-robot systems require a wireless communication network in order to communicate with each other and collaboratively accomplish a given task. A multihop communications network that is self-forming, self-healing and self-organizing is well suited for such mobile robotic systems that exist in unpredictable and constantly changing environments. Because every node in a multihop (or ad hoc) network is responsible for forwarding packets to other nodes, however, the failure of a critical node can result in a network partition. Hence, it may be beneficial to have an ad hoc network configuration that can tolerate temporary failures while allowing recovery. Because movement

of the robotic nodes is controllable, it is possible to achieve such fault tolerant configurations by moving some or all of nodes to new locations.

[0038] Systems and methods consistent with the principles of the invention transform a connected, but non-biconnected network configuration to a biconnected one by suggesting that certain nodes move to new positions. The systems and methods may, for example, place or alter the position of certain nodes such that the newly formed network is more tolerant of node failures.

EXEMPLARY NETWORK

[0039] Fig. 1 is a diagram of an exemplary network 100 in which systems and methods consistent with the principles of the invention may be implemented. Network 100 may include multiple inter-connected nodes 110 N1 - N12. While twelve nodes 110 are illustrated in Fig. 1, a typical network 100 may include more or fewer nodes 110. Nodes 110 may communicate with each other via wireless connections to form a wireless ad hoc network.

[0040] Nodes 110 may include nodes that are capable of moving, such as robotic nodes, and thus, altering the topology of network 100. Each node 110 may operate to perform some function or achieve some objective, either alone or in combination with one or more other nodes 110.

[0041] Nodes 110 may maintain knowledge about the rest of network 100. For example, each node 110 may periodically broadcast a link state update (LSU) that includes its location information (e.g., global positioning system (GPS) coordinates or indoor relative location information) to the rest of network 100. Nodes 110 may use proactive link state-based routing

protocols, such as optimized link state routing (OLSR) and high speed link signaling (HSLS), when sending their LSU.

[0042] Fig. 2 is an exemplary diagram of one of nodes 110 according to an implementation consistent with the principles of the invention. Other ones of nodes 110 may be similarly configured. Node 110 may include a network device 210, a transceiver 220, an antenna 230, and a movement controller 240. Network device 210 may include a robotic device that is capable of moving and performing some function or carrying out some objective, either alone or in combination with other nodes 110. Transceiver 220 may include a conventional transceiver (or a separate receiver and transmitter) that facilitates communication with other nodes 110. Antenna 230 may include a directional and/or an omni-directional antenna that aids in the reception and transmission of data.

[0043] Movement controller 240 may determine the direction and/or distance that node 110 should move, if necessary, to help change the topology of network 100 from a non-biconnected network configuration to a biconnected one. It is important to note that it may not be necessary for all of nodes 110 to move in order to achieve biconnectivity within network 100.

[0044] As will be described in detail below, movement controller 240, according to an implementation consistent with the principles of the invention, may determine the manner (direction and distance or no movement) that node 110 should move to make network 100 biconnected and, thereby, improve the fault tolerance of network 100. Movement controller 240 may be implemented in software, possibly executed by network device 210. Alternatively,

movement controller 240 may be implemented in hardware, or a combination of hardware and software, that provides its determination to network device 210.

[0045] Biconnectivity within a network improves the fault tolerance of the network. For example, if network 100 is biconnected and a node 110 in network 100 fails or becomes unavailable for some reason, network 100 does not become partitioned. Therefore, nodes 110 may attempt to form a biconnected network as long as that does not interfere with their current objective. This may necessitate that some of nodes 110 move to create extra links such that the resultant topology is biconnected.

[0046] Figs. 3A and 3B illustrate a simple example of achieving a biconnected network according to an implementation consistent with the principles of the invention. Fig. 3A is a diagram of a non-biconnected network 310. In network 310, node 6 is a cutvertex, which means that if node 6 were to fail, or otherwise become unavailable, network 310 would become partitioned. In other words, nodes 1 and 2 would no longer be connected to nodes 3-5. Likewise, nodes 3-5 would no longer be connected to nodes 1 and 2. Cutvertex node 6 is shown unfilled in Fig. 3A to contrast it with the other nodes that are shown as filled.

[0047] By moving node 1 (in this example in the direction generally towards the location of node 3), however, non-biconnected network 310 would be transformed to a biconnected network 320, as illustrated in Fig. 3B. In network 320, no cutvertices remain and all nodes are, thus, shown as filled. Therefore, the failure, or unavailability, of any one node would not change the connectivity of network 320.

EXEMPLARY PROCESSING

One-Dimensional Situation

[0048] Fig. 4 is a flowchart of exemplary processing for achieving biconnectivity in a one-dimensional network according to an implementation consistent with the principles of the invention. Figs. 5A and 5B are exemplary diagrams of a one-dimensional network according to an implementation consistent with the principles of the invention. As illustrated in Fig. 5A, all nodes lie in a single line and have only one degree of freedom in movement: they can move either to the right or to the left. Node 5 is a cutvertex in Fig. 5A.

[0049] Processing may begin by determining the initial positions of the nodes (act 410). The nodes may periodically broadcast a LSU that includes its location. The initial positions of the nodes may be represented by $p_i \in R$, $1 \leq i \leq N$, where N is the total number of nodes. Suppose that the positions of the nodes in a new configuration C_{new} are given by $x_i \in R$; $1 \leq i \leq N$.

[0050] A movement schedule for the nodes may then be determined to transform C_{new} to a biconnected configuration, while minimizing the total distance moved by all of the nodes (an isomorphic metric is average distance moved by a node) (act 420). The movement schedule may be represented by:

$$\text{minimize} \qquad D_{total} = \sum_{i=1}^{N} |x_i - p_i|$$

subject to:

$$x_1 \geq p_1; \tag{1}$$

$$x_N \leq p_N; \tag{2}$$

$$x_i - x_{i-1} \geq 0,\ 2 \leq i \leq N; \quad (3)$$

$$x_i - x_{i-2} \leq 1,\ 3 \leq i \leq N. \quad (4)$$

[0051] Constraints 1 and 2 are non-binding constraints that illustrate the fact that the one-dimensional network will compress in length after a biconnected configuration is reached. The N - 1 linear ordering constraints in (3) restrict the search space as no node needs to move past its neighbors to achieve biconnectivity. Biconnectivity is ensured by the N - 2 constraints which basically impose a condition on the nodes that every alternate pairs of nodes are within transmission range of each other. This ensures biconnectivity within the network.

[0052] The movement schedule can be solved optimally in polynomial time (as a function of N). Because the objective function has a non-linear term (absolute value), the movement schedule can be converted into a linear programming (LP) problem in the following manner. N new variables z_i can be introduced to rewrite the optimization problem as follows:

$$\text{minimize} \qquad D_{total} = \sum_{i=1}^{N} z_i$$

subject to:

$$z_i \geq x_i - p_i,\ 1 \leq i \leq N; \quad (5)$$

$$z_i \geq -x_i + p_i,\ 1 \leq i \leq N; \quad (6)$$

$$x_1 \geq p_1; \quad (7)$$

$$x_N \leq p_N; \quad (8)$$

$$x_i - x_{i-1} \geq 0,\ 2 \leq i \leq N; \quad (9)$$

$$x_i - x_{i-2} \leq 1,\ 3 \leq i \leq N. \quad (10)$$

Rewritten in the standard form, this becomes:

minimize $\quad c^T x$

subject to: $Ax \geq b$, where

$$x^T = [x_1, x_2, \ldots, x_N, z_1, z_2, \ldots, z_N]$$

$$c^T = [\underbrace{0, 0, \ldots, 0}_{N}, \underbrace{1, 1, \ldots, 1}_{N}]$$

$$b^T = [-p_1, -p_2, \ldots, -p_N, p_1, p_2, \ldots, p_N, p_1, -p_N, \underbrace{0, 0, \ldots, 0}_{N\text{-}1}, \underbrace{-1, -1, \ldots, -1}_{N\text{-}2}]$$

$$A = \left[\begin{array}{cccccc|cccc}
-1 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 & \ldots & 0 \\
0 & -1 & 0 & \ldots & 0 & 0 & 0 & 1 & \ldots & 0 \\
0 & 0 & -1 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \ldots & -1 & 0 & & & & \\
0 & 0 & 0 & \ldots & 0 & -1 & 0 & 0 & \ldots & 1 \\
\hline
1 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 & \ldots & 0 \\
0 & 1 & 0 & \ldots & 0 & 0 & 0 & 1 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \ldots & 1 & 0 & & & & \\
0 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 1 \\
\hline
1 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\
0 & 0 & 0 & \ldots & 0 & -1 & 0 & 0 & \ldots & 0 \\
\hline
-1 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\
0 & -1 & 1 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & \ldots & -1 & 1 & 0 & 0 & \ldots & 0 \\
\hline
1 & 0 & -1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
0 & 1 & 0 & -1 & \ldots & 0 & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & \ldots & 1 & 0 & -1 & 0 & 0 & \ldots & 0
\end{array}\right]$$

[0053] In the above LP formulation, x and c are $2N$ x 1 vectors, b is a $4N$ - 1 x 1 vector and A is a $4N$ - 1 x $2N$ matrix. In other words, there are $2N$ decision variables (x_i's and z_i's) and $4N$ - 1 constraints. Hence, the problem is amenable to a solution in polynomial time (in N).

[0054] One or more nodes may then move according to the movement schedule (act 430). For example, nodes 1-5 may move to the right in Fig. 5A to transform the network to the network illustrated in Fig. 5B. As shown in Fig. 5B, each node can communicate with the nodes one hop away, as well as the nodes two hops away.

Two-Dimensional Situation: Contraction Technique

[0055] Fig. 6 is a flowchart of exemplary processing for achieving biconnectivity in a two-dimensional network using a contraction technique, according to an implementation consistent with the principles of the invention. Figs. 7A-7C are exemplary diagrams of a two-dimensional network according to an implementation consistent with the principles of the invention. Fig. 7A illustrates an exemplary non-biconnected two-dimensional network. As shown in Fig. 7A, node 6 is a cutvertex.

[0056] Processing may begin with each of the nodes broadcasting a LSU that includes its location information (e.g., GPS coordinates or indoor relative location information) to the rest of the network. When a node receives LSUs from all of the other nodes in the network, the node may extract the location information for the other nodes and determine the network topology therefrom (act 610).

[0057] The node may then determine the geographic center *C* for the entire network (act 620), as illustrated in Fig. 7A. For example, the geographic center *C* may be determined based on:

$$C = \frac{1}{N} \sum_{m=1}^{N} p_m, \qquad (11)$$

where p_m is the position vector of node *m*. In the one-dimensional situation, $p_m \in R$, but in the two-dimensional situation, $p_m = (x_m, y_m)$, $x_m, y_m \in R$.

[0058] After calculating the geographic center *C* of the network, the nodes may determine weighted distances for moving toward *C* (act 630). In this implementation, all nodes move towards *C* by a weighted distance (act 640), as illustrated in Fig. 7B. The weighted distances may be determined based on the following: if a contraction parameter is α, a node *m* with current position p_m may move radially inward towards the center *C* by the distance $(1 - \alpha) \left\| \vec{C} - \overrightarrow{p_m} \right\|$.

[0059] As a result, nodes near the periphery of the network (e.g., nodes 1-5 in Fig. 7B) move a greater distance than the nodes in the interior of the network (e.g., node 6 in Fig. 7B). For example, the nodes very near the center *C* (e.g., node 6) may move very little distance. The rationale behind moving the nodes towards the center *C* is that as they move inward, the network topology will become richer and richer. Also, because of the introduction of more edges (links between nodes), the cutvertices will be removed and the network will eventually become biconnected.

[0060] The choice of parameter α may be important. For example, if $\lim \alpha \rightarrow 1.0$, every node may move only a small distance. On the other hand, if $\lim \alpha \rightarrow 0$, the nodes may eventually

collapse to the center C of the network. Hence, choosing a small α results in unnecessarily dense networks (albeit with higher connectivity than 2), whereas, choosing a large α results in little change in the network topology. In the latter case, the above processing may be repeated until the network is biconnected, as illustrated in Fig. 7C.

[0061] Note that each node m may travel on the same straight line joining its starting position p_m and the center C, even when multiple iterations are needed to make the network biconnected. Hence, when the network reaches a final biconnected configuration with node positions p_m, the total distance traveled may be represented by:

$$D_{total} = \sum_{m=1}^{N} \left\| \overrightarrow{p'_m} - \overrightarrow{p_m} \right\|. \qquad (12)$$

[0062] Two-Dimensional Situation: Block Movement Technique

[0063] Fig. 8 is a flowchart of exemplary processing for achieving biconnectivity in a two-dimensional network using a block movement technique, according to an implementation consistent with the principles of the invention. Figs. 9A and 9B are exemplary diagrams of a two-dimensional network according to an implementation consistent with the principles of the invention. Fig. 9A illustrates an exemplary non-biconnected two-dimensional network. As shown in Fig. 9A, nodes 6 and 7 are cutvertices.

[0064] In this implementation, a biconnected network configuration may be achieved in low order polynomial time while reducing the total distance traveled by the nodes. To do this, all cutvertices may be systematically removed from the network by moving nodes to new locations.

[0065] Processing may begin when the nodes broadcast a LSU that contains their location information. Every node receives LSU updates from other nodes in the network and determines location information regarding the nodes in the network from the LSU updates (act 810).

[0066] The nodes may also extract neighbor information from the LSU updates in order to construct a network graph G (Fig. 9A) that represents a current view of the network topology (act 820). In a perfect world, knowing the location and the transmission range of each node is enough to construct a view of the network topology. In the real world, however, neighbor information from every node may also be useful in constructing the network graph G.

[0067] The network may be represented by the graph $G = (V, E)$ that includes a set of vertices V and a set of edges E, such that $E \subseteq V \times V$. A vertex is a node with a location attribute *pos* and a transmission range attribute R. In one implementation, it may be assumed that all of the nodes include omni-directional antennas and identical wireless propagation characteristics resulting in the same value of R. In this case, an edge $e = (u, v)$ exists between vertices u and v only if $\|u.pos, v.pos\| \leq R$. Graphs, such as graph G, are also known as unit disk graphs. Graph G is called r-partite if V admits a partition into r classes such that every edge has its ends in different classes (i.e., vertices in the same partition class are not adjacent). 2-partite is usually referred to as bipartite.

[0068] A non-empty graph G is called connected if any two of its vertices are linked by a path in G. If $G' \subseteq G$ and G' contains all edges $xy \in E$ with $x, y \in V'$, then G' is a subgraph of G induced or spanned by V' in G. A maximal connected subgraph of G is called a connected component of G. If $A, B \subseteq V$ and $X \subseteq V \cup E$ are such that every $A - - B$ path in G contains a

vertex or an edge from X, then X can be considered to separate the sets A and B in G. A vertex that separates two other vertices of the same connected component is referred to as a cutvertex and an edge separating its endpoints is referred to as a bridge. Thus, the bridges in a graph include those edges that do not lie on any cycle.

[0069] Graph G is called $k-$ connected if $|G| > k$ and $G - X$ is connected for every set $X \subseteq V$ with $|X| < k$. In other words, no two vertices in G are separated by fewer than k other vertices. The greatest integer k, such that G is $k-$ connected, is the connectivity $\kappa(G)$ of G. If $\kappa(G) = 2$, then G is said to be biconnected (i.e., removal of no vertex of G causes a separation of the remaining vertices).

[0070] After constructing a full view of the network topology, each node may independently generate a block tree BT (Fig. 9B) of the topology graph G (act 830). In Fig. 9A, the biconnected nodes (or blocks) of the graph G include nodes 1-5 and 8-12 and the cutvertices include nodes 6 and 7. In Fig. 9B, a corresponding block tree BT is illustrated. It should be understood that Fig. 9B shows only network blocks and not links between nodes within and amongst the blocks.

[0071] The block tree BT may include the following properties:

P1 A block B can have between 0 and N nodes (both inclusive). If two cutvertices are connected by a bridge, then the corresponding block contains no nodes. Block B3 in Fig. 9B illustrates this point. If the original graph has no cutvertices, then it is already biconnected and its block tree includes only one node that contains all N vertices.

P2 A block tree is a bipartite graph. The two classes of the bipartite graph are cutvertices and blocks. No two cutvertices can be adjacent in the block tree, neither can be two blocks.

P3 The block tree is a tree. Because the block tree is bipartite, it cannot have an odd-cycle. The presence of an even cycle would mean that two blocks are connected via two different cutvertices. In that case, one of the two cutvertices can be safely removed without disconnecting the graph. This presents a contradiction and, thus, a block tree is a tree.

P4 A block tree *BT* of a graph *G* can be computed in linear (O (|V| + |E|)) time. This can be achieved during a depth first traversal (DFT), which is a technique generally known to those skilled in the art, of graph *G* in the same pass.

[0072] While executing a DFT process on an undirected graph, an arbitrary node may be chosen as the root. The process may traverse fresh edges of the graph and mark the traversed nodes as "visited." The process may also place identifiers for these nodes onto a stack data structure. The process may continue until a node that is only connected to already visited nodes is reached. At this point, the process may backtrack up to a vertex that has edges connecting it to nodes that have not yet been visited. The node at this vertex will be a cutvertex of the graph. Once a cutvertex has been identified, the process may remove identifiers for the downstream nodes from the stack into a set that corresponds to a biconnected component (called a "block"). Because the above-described acts can be executed during DFT in the same pass, identification of cutvertices and blocks takes only linear time.

[0073] A salient property of a block is that it is a connected subgraph of graph *G*. Therefore, if all nodes in a block are moved together using the same movement vector, distances between all pairs of nodes in that block will remain the same and there will not be any change in the connectivity inside that block. On the other hand, if only one or more nodes in a block are moved, it may result in a change of connectivity within the block. In an implementation consistent with the principles of the invention, all nodes within a block are moved collectively.

[0074] Every node should have the same view of the network topology and, therefore, the same internal representation of the graph *G*. As a result, the DFT process results in the same block tree *BT* at all of the nodes. This can be achieved by a systematic insertion of nodes and edges ordered by node identifiers into graph *G* at every node.

[0075] Suppose a block B_k (not shown) has edges with two cutvertices c_u (not shown) and c_v (not shown) in block tree *BT*. Let blocks B_m (not shown) and B_n (not shown) be two blocks connected to cutvertices c_u and c_v, respectively. In order to minimize the D_{total} metric, the blocks should move as little as possible. If block B_k moves towards block B_m, cutvertex c_u may cease to be a cutvertex but some other node in block B_n may become one as the link between block B_k and cutvertex c_v may be broken. Therefore, no progress may be made towards reducing the number of cutvertices in *G* after this block movement. To prevent this from happening, only blocks in *BT* that have degree 1 may move.

[0076] In order to heuristically minimize the total distance moved, a block that has the maximum number of nodes may be selected as the root block of *BT* (act 840), and identify all other blocks with degree 1 as leaf blocks (i.e., those nodes in the block tree that have only one

emanating edge) (act 850). In the example shown in Figs. 9A and 9B, block B5 may be selected as the root block. Blocks B1, B2, and B4 may be identified as leaf blocks.

[0077] The leaf blocks, and, thus, the nodes within those blocks, may then move to make the network biconnected by systematically removing the cutvertices from *BT* (act 860). In the example of Fig. 9B, all nodes in block B4 may move towards block B5 since block B5 is the parent of block B4. Blocks B1 and B2, on the other hand, have an empty parent block B3. As a result, all nodes in these blocks may move towards the parent cutvertex of the parent block (i.e., cutvertex c2).

[0078] Every block may move in the direction towards the nearest node in the parent block, whenever applicable, by enough distance such that exactly one new edge appears between the current and the parent block. The appearance of this new edge may cause the cutvertex between the two blocks to vanish. Therefore, in one iteration of this processing, several cutvertices may be removed. The time complexity of finding the nearest node as a target edge partner may require B^2 comparisons if B is the average number of nodes in a block. Since $B = O(|V|)$ in the worst case scenario, the process of finding the nearest node has a worst case time complexity of $O(|V|^2)$.

[0079] For large networks, several iterations may be needed to remove layers of cutvertices before only one block remains. Also, after every iteration as the number of blocks increases, the blocks grow in size. Therefore, a small amount of movement by a large block may contribute a significant amount to D_{total}.

[0080] The above processing may be implemented by a MakeBiconnected process performed by movement controller 240 (Fig. 2). The MakeBiconnected process may be represented by:

```
Given: G
G_orig ← G;
BT ← Compute_Biconnected_Components(G);
while (Number_of_Nodes(BT) > 1) do
        MarkRootBlock(BT);
        MarkOtherBlocks(BT);
        Move_Leaf_Blocks(G, BT);
        BT ← nil;
        Recalculate_Edges(G);
        BT ← Compute_Biconnected Components(G);
end while
G is now biconnected;
D_total ← Calculate_Distance_Moved(G_orig, G);
```

The MarkRootBlock(BT) function selects a root block with a maximum number of nodes. The MarkOtherBlocks(BT) function marks leaf blocks and parents. The Move_Leaf_Blocks(G, BT) function may be represented by:

```
Given: G, BT
for all nodes blk ∈ BT do
        if (blk is a BLOCK node and a LEAF) then
                par_cv ← BT.parent [blk];
                par_blk ← BT.parent [par_cv];
                if (Number_of_Nodes(BT[par_blk]) ≠ 0) then
                        nearest ← Find_Nearest_Node(G, blk, par_blk);
                        Translate_Block(BT, blk, nearest);
                else
                        pcv ← BT.parent [par_blk];
                        Translate_Block(BT, blk, pcv);
                end if
        end if
end if
```

[0081] The variable par_{cv} refers to a parent cutvertex and the variable par_{blk} refers to a parent block. The function Find_Nearest_Node(G, blk, par_{blk}) identifies a node in par_{blk} that is nearest from blk. The function Translate_Block(BT, blk, nearest) moves all nodes in blk towards the identified nearest node. The function Translate_Block(BT, blk, pcv) moves all nodes in blk toward the parent cutvertex of par_{blk} (i.e., pcv).

[0082] In the worst case, there may be O (|V|) iterations of the while loop of the MakeBiconnected process before achieving a biconnected configuration (e.g., in the case of a line graph). Because the number of iterations is bounded, however, convergence may be assured in almost all situations except for very special cases. One special case involves the situation where a block's movement causes an oscillation between two points. This special case is illustrated in Fig. 10. This special case can be solved by moving the block towards the nearest node that is a direct parent of the cutvertex. Although doing this repeatedly also guarantees convergence, this process may be used only when movement toward a nearest node in the parent block does not remove a cutvertex.

[0083] There can be two different schemes for moving the nodes: (1) nodes start moving as soon as a single iteration is over, or (2) no node actually starts moving until the final positions of the nodes have been determined (i.e., after convergence of the MakeBiconnected process). Because the convergence occurs rapidly even for large networks, the latter scheme may result in a much lower value of D_{total} due to the vector addition of movement vectors for every node over all iterations of the function.

[0084] Figs. 11A-11D illustrate an exemplary execution of the MakeBiconnected process on a randomly selected initial network topology. The dark points represent cutvertices in the network. Fig. 11A illustrates an initial configuration of the network. There are eleven cutvertices in the initial configuration. Fig. 11B illustrates the network after one iteration of the MakeBiconnected process. Six cutvertices have been removed, leaving five remaining cutvertices. Fig. 11C illustrates the network after a second iteration of the MakeBiconnected process. Now, only two cutvertices remain. Fig. 11D illustrates a final biconnected network configuration. As can be seen, the MakeBiconnected process systematically removes the cutvertices from the network to make the network biconnected after only two iterations.

[0085] While the above-described processing attempts to move leaf blocks only towards their parent blocks or cutvertices in order to remove cutvertices, it may be possible to move leaf blocks towards a non-parent block to remove several cutvertices in a single iteration. Therefore, a more intelligent block movement scheme can reduce D_{total} as well as the number of iterations in suitable situations.

EXEMPLARY SIMULATION RESULTS

[0086] A simulation was performed on a network that included 200 collinear nodes with positions chosen randomly from $[0, 100] \subset R$. Each node was assumed to have a transmission range of 1:0. It was also assumed that all positions corresponding to the initial configuration are known initially. An LP was formulated, as described above, and then optimally solved using matlab.

[0087] Figs. 12A and 12B are graphs that illustrate the total distance moved "D_{total}" and the average distance moved "D_{avg}" by nodes (N) as N is varied. As illustrated in Fig. 12A, D_{total} increases and then decreases in a parabolic fashion as N is increased. One reason for this is because for low values of N, although the initial distances between the nodes is large (owing to uniform random distribution on [0, 100]), there are only a small number of nodes that contribute to D_{total}.

[0088] For large $N \sim 200$, the network is very dense and most of it is already biconnected. As a result, D_{total} is low. The peaks are observed for values of $N \sim 100$ because the network is large and fragmented and many nodes have to move significant distances to achieve biconnectivity.

[0089] As illustrated in Fig. 12B, D_{avg} decreases linearly as N increases. Reasons for this are similar to those given above with regard to changes in D_{total}.

[0090] Another simulation was performed on a network that included a 1 km × 1 km square area with up to 50 nodes randomly distributed therein. All of the nodes were assumed to have omni-directional antennas with transmission ranges of 250 meters each. The ground was assumed to be flat and devoid of obstacles and trenches, thus, allowing the nodes to move anywhere they want. The initial random configuration of nodes may obey a uniform probability distribution while keeping the network connected. One hundred runs for every data point were simulated with the same parameters.

[0091] Figs. 13A and 13B are graphs that compare the performance of the block movement technique against the contraction technique (both described above) with respect to the total

distance moved metric "D_{total}" (Fig. 13A) and average distance moved metric "D_{avg}" (Fig. 13B) while varying the number of nodes (N), and hence the density, in the simulated network. The block movement technique appears to outperform the contraction technique for all values of N considered. One reason for this may be due to the fact that the contraction technique is an ad hoc approach that may unnecessarily move every node, thereby increasing the value of D_{total}.

[0092] As shown in Fig. 13A, the total distance moved increases and then decreases for both techniques, as N is increased from 10 to 50. The reason behind this is that for low values of N, there are only a few nodes that can move and also since the topology is connected, the nodes are not very far from each other. This results in a low value of D_{total}. As N increases, more nodes have to move to make the network biconnected, and this increases D_{total}. As N increases beyond a certain threshold, however, D_{total} begins to drop significantly. This is because higher values of N result in richer, denser topologies that do not have a large number of non-biconnected components. In other words, a little movement causes the topology to become biconnected.

[0093] As shown in Fig. 13B, the curves for the D_{avg} metric look similar to their counterparts for the D_{total} metric apart from the fact that the peaks are shifted slightly to the left owing to division by N.

[0094] Fig. 14 is a graph of an average number of biconnected components (or blocks) in the simulated network while varying the number of nodes (N), and hence the density, in the network. For low values of N, there are only a few blocks and the initial connectivity is sparse. Therefore, a large fraction of blocks have to move in order to make the network biconnected. This results in a high value of D_{avg}. For large values of N, however, the number of blocks reduces as the

network is richly connected at many places, and only a small fraction of blocks needs to be moved to make the network biconnected. This results in low values of D_{avg}. In fact, for N = 50, a node may have to move less than 5 meters on average while following the block movement technique, and about 30 meters on average while following the contraction technique.

[0095] Fig. 15 is a graph of the average number of iterations needed to achieve biconnectivity in the simulated network. As illustrated, the block movement technique requires a lesser number of iterations than the contraction technique. In the contraction technique, if the parameter α is decreased to approximately 0.7–0.8, then a lesser number of iterations would be required. In that case, however, there is a possibility of contracting the network more than necessary. As a result, a high value for α is typically chosen.

[0096] Fig. 16 is a graph of the average diameter of the simulated network while varying the number of nodes. The average diameter may be defined as the maximum length of a shortest path over all source-destination node pairs. The diameter shrinks for all values of N. The diameter is considered a monotonic property in the sense that it can only decrease when edges are added to the network.

CONCLUSION

[0097] Fault tolerance is an extremely desirable property in network design, and biconnectivity is a baseline feature in that domain. Because the position and movement of nodes in an ad hoc network of robotic nodes are controllable, greater fault tolerance can be achieved by moving nodes to locations that result in richer topologies. At the same time, nodes should move as little distance as possible to achieve the desired topological property. Systems and methods

consistent with the principles of the invention provide techniques for moving nodes to new locations, such that the resulting network becomes biconnected.

[0098] The foregoing description of preferred embodiments of the present invention provides illustration and description, but is not intended to be exhaustive or to limit the invention to the precise form disclosed. Modifications and variations are possible in light of the above teachings or may be acquired from practice of the invention.

[0099] For example, the movement controller has been described as being implemented within each of the nodes of the network. In other implementations, the movement controller may be implemented in less than all of the nodes. For example, it may be conceivable that the movement controller in one node provides instructions to one or more other nodes on when and where to move.

[00100] Further, in implementations described thus far, systems and methods have been described for moving nodes within an existing network to achieve biconnectivity. In other implementations, similar systems and methods may be used to form a new network by placing nodes at appropriate locations to achieve biconnectivity. In yet other implementations, similar systems and methods may be used to place new nodes in an existing network to maintain the biconnectivity of the network.

[00101] While series of acts have been described with regard to Figs. 4, 6, and 8, the order of the acts may differ in other implementations consistent with the principles of the invention. Moreover, non-dependent acts may be performed in parallel.

[00102] No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such. Also, as used herein, the article "a" is intended to include one or more items. Where only one item is intended, the term "one" or similar language is used. The scope of the invention is defined by the claims and their equivalents.